Multipath variants for congestion control in context of WebRTC

1st Daniel-George Ardelean

Babeș-Bolyai University
Cluj-Napoca, Romania
daniel.ardelean@stud.ubbcluj.ro

***Abstract*—In a world where the demand for real-time communication is in an ever increasing demand, the need for better performance and lower network costs become imperative in an ever more crowded network space.One of these real-time communication protocols that quickly became a standard for communicating video streams is WebRTC.**

Traditionally communication done with this framework uses a single-path approach, which in the current modern network contest can be improved and will certainly benefit from multipath communication. By taking advantage of the fact that modern network setups can easily include multiple communication paths, research in this field becomes a good point of interest for potential improvements such as: improved resilience, enhanced throughput and bandwidth utilization through aggregation, decreased packet loss and potentially lower latency and jitter.

The aim of this research is to dive deeper into the field of multipath communication for the WebRTC environment, in particular for the congestion control required for this kind of communication. In the current landscape, congestion control algorithms such as GCC, SCReAM or NADA, lack multipath capabilities.

This research proposes a novel wrapper architecture over the NADA algorithm, designed to enable traffic distribution over multiple paths using an array of different strategies. The study utilizes a network simulator to experiment with the potential of this approach and evaluate its possible impact on key performance criteria. The are expected to provide valuable insights into optimizing existing single-path congestion controllers for multipath WebRTC deployments, ultimately contributing to more resilient and high-performing real-time video communication.

***Index Terms*—WebRTC, UDP, Congestion Control, Multipath communication, Networks**

I. Introduction

WebRTC is a framework for real-time communication that operates in a peer-to-peer manner by enabling direct browser-to-browser communication with protocols like RTP [1] that manage media and data flows. However, the dynamic nature of modern networks has frequent path changes, varying bandwidth, and fluctuating delays, as such robust congestion control mechanisms are a must, in order for the usage of such communication frameworks to be as smooth as possible.
The purpose of this paper is to explore the potential benefit of using multipath transmission techniques to improve resilience, throughput, and Quality of Experience (QoE) by leveraging concurrent network paths. As mobile and heterogeneous networks become more common, exploring such alternatives becomes crucial for optimizing real-time performance.
The main reason for WebRTC as the framework of communication is due to two factors: it's usually used for transmitting video data which in itself is a really heavy stream on a network, and due to the fact that the transport protocol used is UDP [2]. A critical aspect for ensuring performance within this protocol is the effective usage of congestion control mechanisms for managing network resources in an effective manner, thereby guaranteeing a smooth data flow and positive user experience by mitigating common issues like packet loss, excessive latency and disruptive jitter [3].
In the face of increasingly complex and congested modern network environments, the concept of multipath strategies emerged as a promising strategy to augment congestion control. By utilizing multiple network pathways, applications can achieve enhanced reliability through redundancy, increased throughput via bandwidth aggregation, and a more efficient utilization of available network resources. The current state of the art in the context of WebRTC lacks any significant depth into the topic of multipath routing on top of existing congestion control algorithms. The purpose of this paper is to explore this multipath approach over UDP.

II. Background

WebRTC was initially developed to create a simple real time connection between different browser to enhance sending data, but it quickly expanded its initial use cases with significant a importance in video conferencing which lead to an encompass to areas like telemedicine, gaming and IoT. This expansion quickly translated into a broader set of requirements for congestion control algorithms which leads to a really high level of adaptability to different application needs and varying network conditions.
WebRTC architecture is designed to facilitate direct, peer-to-peer communication between web browsers, primarily using the User Datagram Protocol (UDP) which by itself does not inherently provide any congestion control mechanisms within WebRTC [4]. Initial research in WebRTC congestion control focused on single-path approaches, tailoring mechanisms to suit interactive media sessions. These mechanisms generally involve feedback loops based on packet loss, delay, and jitter to adapt the sending rate. Modern WebRTC congestion control algorithms combine these loss and delay signals to achieve a hybrid approach that is more robust and responsive adaptation

[4].
Integration of these types of feedback allows these algorithms to not only react to existing congestion (loss) but also anticipate potential congestion (increasing delay) leading to more effective rate adjustments. More recently, researchers have highlighted the need for more dynamic approaches to handle scenarios where multiple concurrent paths are available, such as in dual connectivity scenarios or when using overlay networks.
One of the more prevalent issues that happened with WebRTC was its extremely fast penetration and acceptance rate, which led to a quick loss of control over the standardization of the various systems it was integrated in. A thorough process was conducted by the group RTP Media Congestion Avoidance Techniques (RMCAT) and the processes are thoroughly explained in [4]. In their research several open issues were underscored including: ensuring fairness across flows, adapting quickly to sudden network changes, and managing the interactions between media and data flows. Multipath strategies promise to address some of these challenges by distributing the load and offering redundancy.

III. Congestion control

A network is categorized as congested when the volume of data traversing a network exceeds the network's capacity to handle it efficiently. This overload leads to a degradation in the quality of service experienced by users [3]. Such situations become a common occurrence in networks where numerous devices and applications compete for a limited transmission capacity.
Congestion control techniques in the environment of WebRTC can be classified in two categories: open-loop and closed-loop. Open-loop congestion control is realized as a set of proactive techniques that prevent network congestion before it occurs. These methods operate by implementing various policies at the source or destination, aiming to avoid overwhelming the network. Among these policies numerous methods are employed such as: packet retransmission, acknowledgement policies, discarding policies or admission control mechanisms. A common thing all these methods have is the fact that they start keeping track of a reference of some kind: a suspicion that a packet was lost or corrupted packets, acknowledging N packets which determines the sender's transmission rate, if a packet has a really low importance it's automatically discarded when the queue starts filling up.
Closed-loop in contrast involves techniques that react to congestion after it has occurred. This is achieved through feedback mechanisms within the network to detect and alleviate that congestion. There are a number of techniques applied for this type of congestion: destination sends a choke packet back to the source to inform it about the congestion, backpressuring through the network from node-to-node to slowly propagate the information that the network is congested, directly signaling the congestin or some more indirect signaling approaches through various factors such as lack of acknowledgements or increased round-trip times. Usually routers implement by default congestion avoidance mechanisms to proactively manage traffic and prevent network collapse.
In particular, the direction for this paper is focused on real-time media streams over RTP/UDP. For this particular setting, the algorithms which are the most prominent are: NADA (Network-Assisted Dynamic Adaptation) [5], SCReAM (Self-Clocked Rate Adaptation for Multimedia) [6], GCC(Google Congestion Control Algorithm) [7]. Each of these algorithms has its own special use-case and problem that it's solving: SCReAm excels in wireless (LTE) environments by quickly reacting to capacity drops within one RTT and maintaining throughput, GCC yields low queueing delay under light congestion and demonstrates fairness against TCP flows and NADA achieves smooth rate adjustments and can prioritize flows via weighted control gains, preventing starvation and offering stability across mixed networks.
After careful considerations, the NADA algorithm was decided to be the most suited for the current research due to its flow prioritization which will become useful for approaching a multipath variant.

IV. Multipath communication

In the context of networks, multipath communication refers to the networking technique of establishing and utilizing multiple distinct paths between a sender and a receiver. This approach allows data to be transmitted in concurrent manner over these paths, strategy which naturally leads to several advantages over the traditional single-path communication.
Establishing this topology of multiple paths, facilitates the communication to be more resilient to network failures and congestion by seamlessly continuing the communication over the other available paths ensuring availability and a more stable connection. By simultaneously transmitting data over multiple paths, the aggregate bandwidth available for the communication can be increased, leading to faster data transfer rates and a reduction in overall latency. This distribution of traffic will also optimize the use of available bandwidth leading to more efficient operations. And besides all these advantages one of the most potent ones comes as an aid for mobile devices which on their own often have multiple network interfaces like Wi-Fi and cellular. A multipath communication enables seamless handover between these interfaces which allows for a communication without interruption providing a more consistent and reliable user experience. [8]
All these improvements prove that multipath communication is quite an offering framework that is guaranteed to bring enhancements to applications that take advantage of it such as: increased reliability, increased throughput, improved resource utilization, better load balancing across the network and seamless handover.
This multipath approach is present in the fourth layer of the OSI model, the Transport layer [9], and mutates the existing protocols TCP and UDP into some quite unique approaches. The Multipath TCP (MPTPC) enables TCP to simultaneously use multiple paths by establishing subflow across the available paths. This approach enables the aggre-

gation of bandwidth from multiple network interfaces leading to significantly higher data transfer rates than what a single path could provide [10] as well as the other advantages listed when defining the multipath approach. While this MPTCP is a robust solution for multipath communication, in the context of WebRTC we cannot afford the reliance of a TCP connection. This orients the research to a multipath UDP (MPUDP) approach. While this protocol does not inherently support multipath capabilities, it still accepts some workarounds to take advantage of multiple paths in a network.

One such development is the Multipath Datagram Congestion Control Protocol (MP-DCCP). This particular implementation is sensitive to latency and may have varying demands for reliability and in-order delivery. This approach operates the IP layer providing a framework for flexible traffic steering, switching and splitting across multiple paths [11]. Other attempt is the Mobile Shell (Mosh), which operates at the application layer that UDP utilizes, to select a dynamic address to improve reliability and performance over lossy and high-latency links. Mosh can smartly switch between available network paths based on their quality providing a more seamless user experience in mobile environments [12].

Despite the presented advantages there still exist quite a lot of challenges such as: the required sophisticated scheduling algorithms for such an approach, handling out of order packets due to the varying nature of different paths and presence of middleboxes such as NAT routers or firewalls that might not be compatible with such a strategy might not allow for a multipath UDP approach.

When it comes to multipath for WebRTC traffic, one particular aspect that can be used is the support for multiple ICE candidates. Each of these peers gathers a set of candidate transport addresses, where each candidate pair represents a potential network path for communication. In standard WebRTC only one of these pairs is nominated to be used for the session, however the underlying gathering of possible paths of the ICE framework can be a good starting research point to be used as a primary connection.

The problem for this approach for WebRTC comes down again to how the packets might arrive out of order, problem which reinforces the need for a robust implementation of a reordering mechanism at the receiver to make sure the media stream is reconstructed correctly, as well as an effective load balancing strategy for choosing the paths, strategy that should take into consideration the paths individual capacities and the current network conditions.

V. Approach

When it comes to the actual practical aspect of this research, the results were extracted using a network simulator, ns-3 [13].This simulator allows writing scenarios of different network topologies and sending packets on them. Usual classes such as routers, TCP or UDP clients, peer-to-peer connections between routers and other common utilities are already implemented in the simulator. For more complex algorithms such as NADA [5], a custom UDP client wrapper had to be created, as well as simulated video traffic, as the simulator does not inherently support sending video packets. After having implemented these baseline necessities, the topology of the network simulation had to be implemented. For the single path approach there exists a source, a destination and an intermediary router. On the intermediary router there can exist multiple competing sources of traffic to emulate what a normal router can experience on the internet.

Although ns-3 offers a wide range of helper classes, they are mostly limited to primitives guaranteed to exist in any real-world network. To implement the proposed solution, it was necessary to create custom wrapper classes around the default UDP clients, implement our own simulated video packet formats, and develop an application on the destination side to emulate a video stream consumer.

With a clear understanding of the existing congestion control algorithm ecosystem, we decided to use the NADA algorithm due to its hybrid nature and average performance. It does not overly excel in a single metric at the cost of others, offering a neutral middle ground that allows us to focus on implementing multipath communication methodologies and the related strategies. The NADA algorithm is implemented in accordance with its RFC specification [5].

A. Path selection strategies

Before formally describing each strategy, it's important to understand the principles behind their design. Each strategy is composed of two principles:

1) calculating scores for each network path to create a weight and assign it
2) adapting to the network's dynamic conditions

The score for each path is based on the NADA transmission rate (rateScore) 1, as well as: packet utilization rate (ratio of acknowledged to sent_packets, utilized_packets) 3, and the RTT on that path (rttScore) 2. The final qualityScore is computed as shown in 4, and is used by almost all strategies (except round-robin) to determine the final weight assigned to each path.

$$\text{rateScore} = \frac{1}{1 + \frac{\text{RTT}}{100}} \tag{1}$$

$$\text{rttScore} = \frac{\text{rata_NADA}}{\text{bandwidth_mean}} \tag{2}$$

$$\text{utilized_packets} = \frac{\text{acked_packets}}{\text{sent_packets}} \tag{3}$$

$$\begin{aligned}\text{qualityScore} = {} & 0.3 \times \text{rttScore} + 0.4 \times \text{rateScore} \\ & + 0.3 \times \text{utilized_packets}\end{aligned} \tag{4}$$

1) Dynamic Weights: This strategy takes each paths qualityScore and calculates a weight based on normalization against the total quality of the network (sum of all qualityScores), and then applies a smoothing transition to avoid aggressive fluctuations 5. This makes it suitable for environments where

path performance is comparable and the goal is to maximize total throughput.

$$\text{dynamic_weight} = 0.7 \times \text{currentWeight} + 0.3 \times \text{normalizedWeights} \quad (5)$$

2) Best Path: This strategy also relies on the qualityScore 4 and compares the scores across available paths. The best-performing path gets 80% of the traffic, while the other gets 20%. Keeping a minimal traffic flow on the alternate path serves as a backup and continuous monitoring method, allowing rapid switching in case the primary path degrades. Ideal when one path is clearly superior.

3) Round Robin: Traffic is equally distributed across all paths. No dynamic weight adjustment is applied—each packet is sent in turn over a different path. This strategy serves as a control group, directly comparable with **AggregatePathNada-Client** as it mimics its multipath sending behavior.

4) Frame Aware: This strategy adjusts path selection dynamically based on video frame characteristics. Keyframes (isKeyFrame = true) are prioritized and sent over the path with the lowest RTT for reliability, while delta frames (isKeyFrame = false) are distributed using a weighted load-balancing formula that considers both bandwidth and latency.

$$\text{normalizedWeight} = \frac{\text{rateMbps}}{\text{rttMs}} \times \frac{1}{\text{totalMetric}} \quad (6)$$

5) Buffer Aware: Similar to the previous strategy, this one aims to optimize the end-user experience by ensuring the receiver's video buffer always has content, avoiding freezes. If buffer levels drop below 50%, the system enters "emergency mode" where the lowest RTT path gets 90% of the traffic. Otherwise, path weights are calculated as shown in 7. The **bufferWeightFactor** (default 0.3) defines the relative importance of buffer status vs. path quality. **bufferWeight** is calculated using urgency factors that favor low-RTT paths when the buffer is close to empty, while **qualityWeight** uses previously calculated quality scores. This ensures balanced traffic under normal conditions and prioritizes quick, reliable paths when buffer levels threaten playback continuity (to prevent rebuffering).

$$\text{combined_weight} = \text{bufferWeight} \times \text{bufferWeightFactor} + (1 - \text{bufferWeightFactor}) \times \text{qualityWeight} \quad (7)$$

VI. Simulations topology

To balance complexity and clarity, a simplified yet effective network topology was selected. A source node transmits video traffic to a destination through two independent paths: Path A (via RouterA) and Path B (via RouterB), each with distinct bandwidth and latency characteristics. This asymmetric setup emulates real-world network conditions where multiple routes exhibit varying performance due to infrastructure, routing policies, or ISP diversity 1.

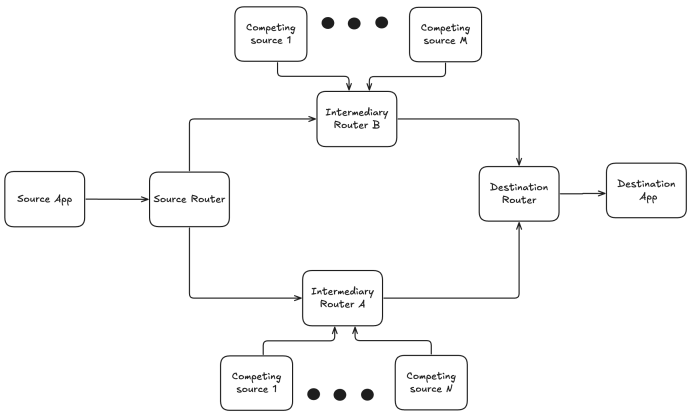

Fig. 1. Simulation topology

Each path includes configurable TCP sources that generate background traffic using the BulkSendHelper application and the TCP NewReno congestion control algorithm [14]. These sources operate on separate IP subnets (10.11.x.0 for Path A, 10.21.x.0 for Path B), ensuring correct routing and traffic isolation.

The use of concurrent TCP flows creates realistic conditions of statistical multiplexing, where independent traffic sources share network resources dynamically. TCP's congestion control, based on AIMD (Additive Increase/Multiplicative Decrease), naturally produces traffic fluctuations through phases like slow start, congestion avoidance, and recovery after packet loss.

This setup challenges NADA to fairly compete with TCP traffic, validating its coexistence in mixed network environments. Moreover, the variability in the TCP flows differing start/stop times, RTTs and load, introduces dynamic congestion patterns that test the adaptability of NADA beyond static scenarios.

Adjustable flow intensity allows simulation of both aggressive and conservative applications, supporting detailed analysis of NADA performance under various congestion levels.

VII. Network traffic

Both simulations use emulated video packets that accurately reflect real-world streaming behavior. The video transmission model is frame-based, with a hierarchical structure of key frames and delta frames—key frames contain full image data and appear every 100 frames, while delta frames encode only incremental changes. Key frames are typically 1.5–2X larger than base frames, following modern video coding standards like H.264 and H.265.

Frames are fragmented into MTU-sized packets (typically 1000–1400 bytes) using equation 8. Each fragment includes sequence and timing info to ensure proper reassembly.

An adaptive rate control mechanism monitors network conditions via the NADA algorithm and dynamically adjusts frame sizes—lower quality under poor conditions, higher quality

when bandwidth allows mimicking real-world video encoders that balance quality and bitrate.

$$\text{numPackets} = \frac{\text{frameSize} + \text{mtu} - 1}{\text{mtu}} \tag{8}$$

VIII. Video Receiver

The **VideoReceiver** class simulates a real-time video streaming client by implementing a complete buffer management system, frame reconstruction, and adaptive playback. It uses a UDP socket to receive packets and processes them via a callback.

Playback starts after a pre-buffering phase, where a few frames (default: 5) are accumulated. The initial buffering time is calculated as: $max(10, initialBufferFrames \times m_frameInterval.GetSeconds())$,this buffer performance being monitored every 100ms.

The system uses two structures: m_incompleteFrames for assembling frames and m_frameBuffer for ready-to-play frames. Key frames are complete after ≈ 3 packets, delta frames after ≈ 2. A frame is considered complete if it has 90% of the expected data. Frames are consumed every $\frac{1}{frameRate}$ seconds, simulating decoding and display. For each frame, end-to-end delay is recorded based on the arrival time of its first packet. Adaptive mechanisms handle underruns and adjust playback:

- After 3 underruns, rebuffering time increases by $500 + (underruns - 3) \times 100ms$
- If $buffer < 2$ frames $\rightarrow$ next interval increases by 10%
- If $buffer > 10$ frames $\rightarrow$ interval decreases by 5%

Each UDP packet includes metadata (frame type, ID, sequence). Frames are reassembled using NADA sequence numbers, which encode both frame and packet indices. Late frames (older than 10 frames behind the latest) are discarded to prevent buffer congestion under high packet loss.

IX. Metrics

This research focuses on both network performance metrics and application-level quality indicators, offering a comprehensive view of how multipath congestion control impacts real-time video streaming and user experience.

A. Throughput

Represents the effective data rate, calculated as the total received bytes divided by simulation time, expressed in Mbps (see Equation 9). It reflects bandwidth utilization efficiency—especially in multipath setups—and directly impacts video quality (e.g., resolution, frame rate). Low throughput leads to aggressive compression, artifacts, pixelation, and reduced video clarity.

$$\frac{\text{rxBytes} \times 8.0}{\text{simulationTime}} \times \frac{1}{1{,}000{,}000} \tag{9}$$

B. End-to-End Delay

Measures the total time for a packet to travel from source to destination, including propagation, queuing, and processing delays. It is computed as:

$$\frac{\text{delaySum.GetSeconds()}}{\text{rxPackets}}$$

,low delay ensures smooth interaction in real-time apps, while high delay can cause noticeable lag and increased buffer requirements.

C. Packet loss

Quantifies the percentage of packets sent but not received:

$$100.0 \times \frac{\text{txPackets} - \text{rxPackets}}{\text{txPackets}} \tag{10}$$

Loss impacts video quality severely, especially for key frames (I-frames), causing freezes, artifacts, and desynchronization. Rates above 2–5% are typically very noticeable.

D. Jitter

Represents the variability in packet arrival times and is calculated as 11

$$\frac{\text{jitterSum.GetSeconds()}}{\text{rxPackets} - 1} \tag{11}$$

High jitter disrupts playback and synchronization, requiring larger buffers, which may in turn increase latency.

E. Buffer Length

Indicates how many seconds of video are preloaded in the playback buffer. A sufficiently full buffer ensures smooth playback during transient drops in network quality. However, overly large buffers can delay video start or interaction responsiveness.

F. Buffer Underruns

Occurs when the buffer is emptied before new data arrives, causing playback interruptions. This is a critical metric for assessing whether the transmission can keep up with the playback rate.

G. Mean Opinion Score (MOS)

An estimated score (0–5) based on the bitrate, mapped via a logistic function. It approximates perceived visual quality:

- 4-5: Excellent
- 3-4: Good
- <3: Poor experience

MOS integrates bitrate stability and responsiveness to network conditions.

H. Throughput Stability (Standard Deviation)

Measures how consistent the bitrate is over time. Sudden drops or spikes can lead to quality changes and playback disruptions. In multipath setups, this indicates how well traffic is balanced across paths.

I. Energy Efficiency

Expresses the ratio of successfully delivered bytes to total transmitted bytes 12

$$\frac{\text{successfully_delivered_bytes}}{\text{total_transmitted_bytes}} \times 100 \tag{12}$$

It highlights the system's ability to deliver good video quality with minimal energy, especially important for mobile or battery-powered devices.

J. Improvement

Due to the large quantity of experiments and results, an aggregated metric was created to quantify the positive outcomes. The improvement metric is calculated as the sum of differences in improvement for each of the above metrics.

X. Experiments

The experiments aimed to comprehensively evaluate the performance of the proposed Multipath NADA system compared to traditional Aggregated NADA. Ten distinct network scenarios were designed, each simulating realistic conditions with varying bandwidth, latency, and congestion levels.

All simulations were conducted using NS-3 version 3.44, with dedicated modules for NADA and multipath traffic handling.

Each base scenario was tested using five multipath path selection strategies (see Section V-A) and compared against a single-path NADA client. The experimental scenarios are detailed in I

Simulation outputs were timestamped and preserved in raw format to support later analysis. Visualization of key metrics was performed using matplotlib [15] and seaborn [16], aiding in the quick identification of trends or anomalies.

Overall, this experimental framework ensures reproducible and meaningful performance insights for Multipath NADA across varied network conditions.

XI. Results

Multipath NADA significantly improves throughput across all scenarios. The Weighted Best Path strategy performs best, especially in legacy broadband networks, with gains up to 176.8%. Notable improvements also appear in Enterprise Multi-Gigabit and Data Center scenarios, confirming the scalability of the approach 2.

Multipath strategies reduce latency modestly but consistently (0.3%–8.2%)4. Legacy broadband sees the biggest gains. Round Robin and Frame Aware strategies offer the most consistent improvements, making them suitable for low-latency applications 3.

Packet loss is reduced drastically in affected scenarios (88.9%–99.9%), especially in legacy broadband. However, in hybrid Satellite + Terrestrial setups, some adaptive strategies may worsen loss, highlighting the need for better adaptation mechanisms 5.

Results vary: jitter drops by up to 86.7% in high-speed research networks and by $\tilde{7}0\%$ in data centers. However,

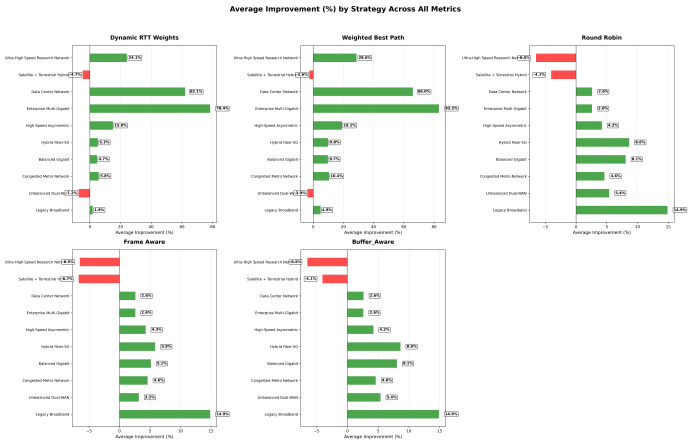

Fig. 2. Improvement by multipath strategy

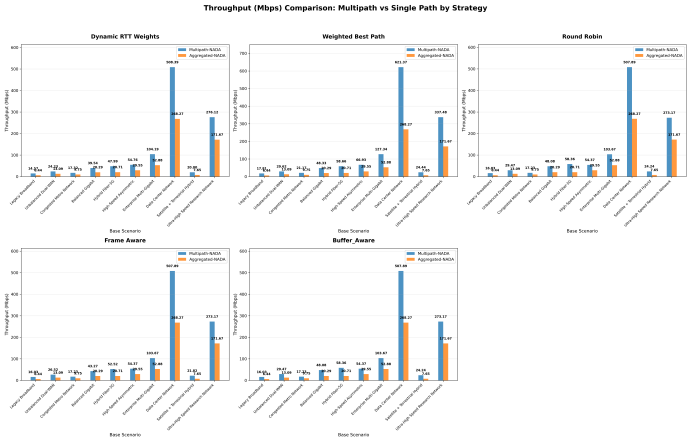

Fig. 3. Throughput results

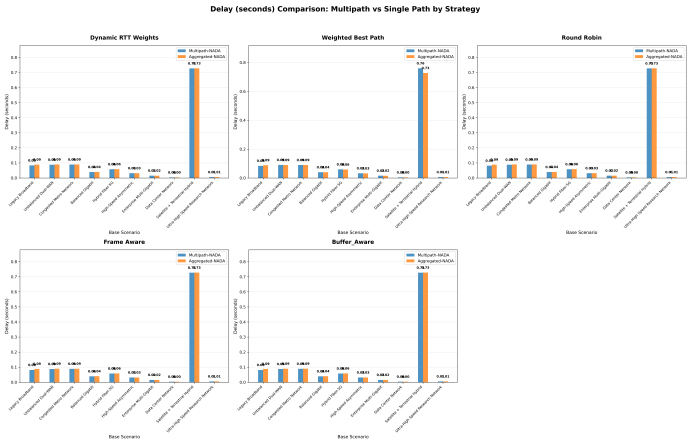

Fig. 4. Delay results

TABLE I
EXPERIMENTAL SCENARIOS FOR EVALUATING MULTIPATH NADA PERFORMANCE

Scenario Name	Bottleneck Bandwidth	Path 1 Rate	Path 2 Rate	Delay Path 1 (ms)	Delay Path 2 (ms)	Frame Rate (fps)	Src A (Loss)	Src B (Loss)	Sim. Time (s)
Legacy Bandwidth	100 Mbps	60 Mbps	50 Mbps	35	50	30	8 (0.8)	6 (0.85)	60
Unbalanced Dual WAN	200 Mbps	200 Mbps	50 Mbps	20	80	60	10 (0.75)	8 (0.9)	60
Congested Metro Network	300 Mbps	500 Mbps	500 Mbps	40	50	60	12 (0.85)	12 (0.85)	60
Balanced Gigabit	1 Gbps	600 Mbps	600 Mbps	18	22	60	10 (0.8)	10 (0.8)	60
Hybrid Fiber-5G	1.2 Gbps	1 Gbps	800 Mbps	10	45	60	8 (0.7)	12 (0.85)	60
High-Speed Asymmetric	2.5 Gbps	2 Gbps	1.2 Gbps	10	25	90	10 (0.8)	8 (0.8)	50
Multi-Gigabit Corporate	5 Gbps	3 Gbps	3 Gbps	6	10	120	9 (0.75)	9 (0.75)	45
Data Center Network	10 Gbps	6 Gbps	6 Gbps	1	3	120	8 (0.8)	8 (0.8)	40
Hybrid Satellite + Terrestrial	1 Gbps	800 Mbps	600 Mbps	20	650	60	8 (0.7)	12 (0.9)	60
Ultra-Fast Research Network	20 Gbps	15 Gbps	12 Gbps	2	4	120	7 (0.7)	7 (0.7)	35

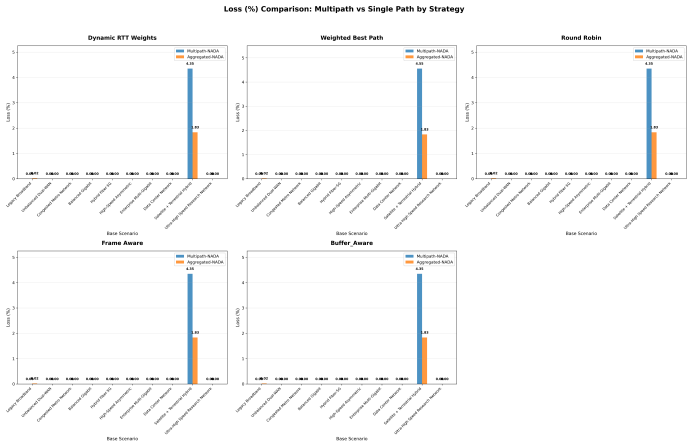

Fig. 5. Packet loss results

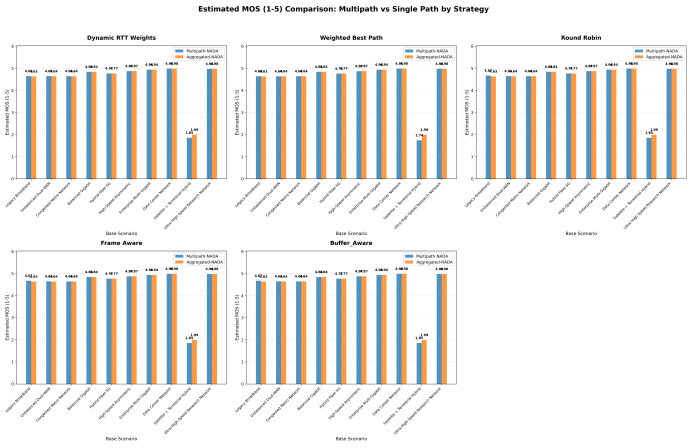

Fig. 6. MOS results

in legacy broadband, adaptive strategies increase jitter (up to 106%), while simpler ones like Round Robin and Buffer Aware show more stable results (64.6%–90.3% improvement) 2.

Despite higher throughput, buffer lengths are generally lower in multipath simulations. This is due to smarter packet delivery. Buffer underruns are significantly reduced (26.7%–100%), indicating stable video playback.

User experience improves slightly but consistently (0.01%–0.88%), with the largest gains in legacy broadband.

All multipath strategies show increased throughput variability (84.3%–218.5%), especially in broadband networks. This suggests a need for smoothing techniques, such as filtering or hysteresis-based path switching.

An overall score for each metric was calculated as a sum of the differences in performance for each metric. These results we're placed in the heatmap represented in 7

XII. CONCLUSION AND FUTURE DIRECTIONS

This research clearly demonstrates that applying multipath strategies to the NADA congestion control algorithm significantly enhances performance in real-time video streaming. The Weighted Best Path strategy stood out, improving throughput by 60.8% to 219.4% across various network types. These gains reflect the adaptability of multipath systems to diverse infrastructures.

Packet loss reductions between 88.9% and 99.9% and the near elimination of buffer underruns translated into much smoother video playback. While improvements in MOS scores were modest (up to 0.88%), they consistently indicated better user experience.

However, increased throughput variability (up to 218.5%) reveals a trade-off that higher capacity often comes with less

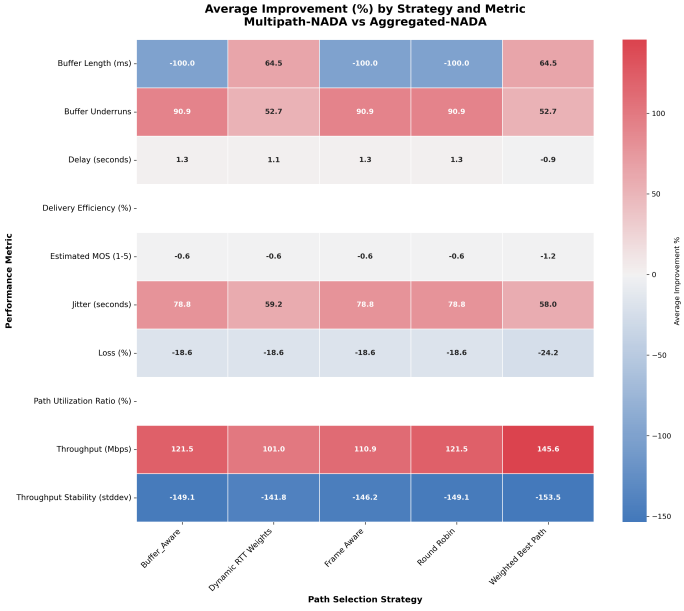

Fig. 7. Metric performance heatmap

predictability. Similarly, jitter improvements were significant in high-performance networks but worsened in unstable broadband environments. Path utilization also remained low in most scenarios (below 20%), indicating untapped potential.

Overall, the results show that combining a dominant stable path with additional bandwidth through less stable paths (as in Weighted Best Path) offers a hybrid approach that balances robustness with performance.

When it comes to future directions, several promising avenues for further exploration in multipath video transmission. A key priority is the development of an intelligent adaptive system for strategy selection, based on machine learning algorithms capable of optimizing real-time performance. Another essential direction focuses on advanced rate stabilization mechanisms through adaptive filtering, hysteresis, and intelligent buffer management. Optimizing path utilization via proactive traffic distribution and congestion prediction can enhance system efficiency.

Hybrid strategies that combine the robustness and performance of different methods may yield significant benefits in variable network conditions. It is also important to extend testing to emerging scenarios such as 5G/6G, edge computing, IoT, and low-latency applications, alongside integration with modern codecs and adaptive streaming techniques.

Evaluating user perception is necessary to correlate technical metrics with subjective experience. Finally, standardizing evaluation methodologies and validating solutions in real-world physical networks are critical steps toward practical applicability.

This work provides strong evidence that multipath transport is a viable and impactful enhancement for real-time video delivery. It lays a solid foundation for future communication systems that are smarter, more adaptable, and more responsive to user needs. In a world moving toward 5G, 6G, and immersive applications, such multipath strategies could become essential for achieving high-quality, resilient streaming.

References

[1] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Rtp: A transport protocol for real-time applications," Tech. Rep., 2003.

[2] "User Datagram Protocol," RFC 768, Aug. 1980. [Online]. Available: https://www.rfc-editor.org/info/rfc768

[3] B. Jansen, T. Goodwin, V. Gupta, F. Kuipers, and G. Zussman, "Performance evaluation of webrtc-based video conferencing," *ACM SIGMETRICS performance evaluation review*, vol. 45, no. 3, pp. 56–68, 2018.

[4] L. De Cicco, G. Carlucci, and S. Mascolo, "Congestion control for webrtc: Standardization status and open issues," *IEEE Communications Standards Magazine*, vol. 1, no. 2, pp. 22–27, 2017.

[5] X. Zhu, R. P. *, M. A. Ramalho, and S. M. de la Cruz, "Network-Assisted Dynamic Adaptation (NADA): A Unified Congestion Control Scheme for Real-Time Media," RFC 8698, Feb. 2020. [Online]. Available: https://www.rfc-editor.org/info/rfc8698

[6] I. Johansson and Z. Sarker, "Self-Clocked Rate Adaptation for Multimedia," RFC 8298, Dec. 2017. [Online]. Available: https://www.rfc-editor.org/info/rfc8298

[7] S. Holmer, H. Lundin, G. Carlucci, L. D. Cicco, and S. Mascolo, "A Google Congestion Control Algorithm for Real-Time Communication," Internet Engineering Task Force, Internet-Draft draft-ietf-rmcat-gcc-02, Jul. 2016, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-rmcat-gcc/02/

[8] M. Li, A. Lukyanenko, Z. Ou, A. Ylä-Jääski, S. Tarkoma, M. Coudron, and S. Secci, "Multipath transmission for the internet: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2887–2925, 2016.

[9] M. M. Alani and M. M. Alani, "Osi model," *Guide to OSI and TCP/IP Models*, pp. 5–17, 2014.

[10] S. Barré, C. Paasch, and O. Bonaventure, "Multipath tcp: from theory to practice," in *NETWORKING 2011: 10th International IFIP TC 6 Networking Conference, Valencia, Spain, May 9-13, 2011, Proceedings, Part I 10*. Springer, 2011, pp. 444–457.

[11] M. Amend, A. Brunstrom, A. Kassler, V. Rakocevic, and S. Johnson, "DCCP Extensions for Multipath Operation with Multiple Addresses," Internet Engineering Task Force, Internet-Draft draft-ietf-tsvwg-multipath-dccp-24, Apr. 2025, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tsvwg-multipath-dccp/24/

[12] M. Boutier and J. Chroboczek, "User-space multipath udp in mosh," *arXiv preprint arXiv:1502.02402*, 2015.

[13] A. B. Massimo Antonio Tozzato, Fabio Cazzin. (2025) Network simulator 3. [Online]. Available: https://www.nsnam.org/

[14] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, Apr. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6582

[15] (2025) Matplotlib. [Online]. Available: https://matplotlib.org/

[16] (2025) Seaborn. [Online]. Available: https://seaborn.pydata.org/